Measures for Software Product Lines: A White Paper for the Office of the Undersecretary of Defense, Science and Technology, Software Engineering

April 2001

Dave Zubrow
Software Engineering Institute
Carnegie Mellon University

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 01-04-2001	2. REPORT TYPE	3. DATES COVERED (FROM - TO) xx-xx-2001 to xx-xx-2001

4. TITLE AND SUBTITLE Measures for Software Product Lines: A White Paper for the Office of the Undersecretary of Defense, Science and Technology, Software Engineering Unclassified	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Zubrow, Dave ;	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME AND ADDRESS Booz Allen & Hamilton 8283 Greensboro Drive McLean, VA22102	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS Software Engineering Institute ,	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This paper is one of a series commissioned by the Department of the Undersecretary of Defense, Science and Technology (DUSD) (S&T) that characterizes the status of measurement associated with a particular aspect of software engineering. The specific focus of this paper is measures for software product lines.

15. SUBJECT TERMS
IATAC Collection; software engineering; software product lines

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Public Release	18. NUMBER OF PAGES 22	19. NAME OF RESPONSIBLE PERSON Fenster, Lynn lfenster@dtic.mil
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39.18

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 074-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503			
1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** 4/1/2001	**3. REPORT TYPE AND DATES COVERED** Report 4/1/2001	
4. TITLE AND SUBTITLE Measures for Software Product Lines: A White Paper for the Office of the Undersecretary of Defense, Science and Technology, Software Engineering			**5. FUNDING NUMBERS**
6. AUTHOR(S) Dave Zubrow			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Booz Allen & Hamilton 8283 Greensboro Drive McLean, VA 22102			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Software Engineering Institute			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; Distribution unlimited			**12b. DISTRIBUTION CODE** A
13. ABSTRACT *(Maximum 200 Words)* This paper is one of a series commissioned by the Department of the Undersecretary of Defense, Science and Technology (DUSD) (S&T) that characterizes the status of measurement associated with a particular aspect of software engineering. The specific focus of this paper is measures for software product lines.			
14. SUBJECT TERMS IATAC Collection, software engineering, software product lines			**15. NUMBER OF PAGES** 21
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UNLIMITED

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

1. Overview

This paper is one of a series commissioned by the Department of the Undersecretary of Defense, Science and Technology (DUSD) (S&T) that characterizes the status of measurement associated with a particular aspect of software engineering. The specific focus of this paper is measures for software product lines.

Software product lines are receiving growing attention as a potential development approach. The concept addresses many problems noted today and provides solutions sought by software developers and acquirers alike. Some of the problems addressed by a product line approach include:

- dissatisfaction with current project/product performance
- need to reduce cost and schedule
- complexity of managing and maintaining too many product variants
- lack of staff
- need to quickly respond to customer / marketplace demands

Some of the benefits expected from a software product line include the:

- ability to deliver products faster, better, and cheaper
- ability of marketing agents to sell features (in addition to products)
- ability to respond to changing requirements during development
- ability to better manage and maintain product variants

This paper begins by defining and describing the characteristics of a software product line to establish a context for measurement. The second section describes the evolution of Department of Defense (DoD) interest in software reuse and current DoD policies as they relate to software product lines. The third section describes the information needs to be addressed with software measures and the fourth section proposes some product line measures and discusses the current status of the measures in terms of their practical application. The final section offers recommendations for next steps to improve the development and use of measures to be used in the acquisition and development of systems based on a software product line. In summary, this investigation found little has been published on software measurement in the context of software product lines; although, there is considerable work on software measurement (e.g. software reuse) that is relevant to software product lines. Therefore, the recommendations for software measures included in this paper are based on the experience of the use these measures in contexts other than a software product line. Nonetheless, the recommended set of measures should provide adequate insight into the operation of a software product line.

2. Software Product Line Definition

A software product line is a *set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission.* Substantial economies can be achieved when the systems in a software product line are developed from a common set of core assets. However, building or acquiring a software product line and bringing it to market requires a blend of skillful engineering and both technical and organizational management to overcome the pitfalls that may bring disaster to an unsophisticated organization [Clements 99].

As described in the SEI's Software Product Line Framework [Clements 99], a software product line involves three spheres of management responsibility (See Figure 1):

1. **product line** : the overall business endeavor of the enterprise
2. **asset development**: the development and sustainment of reusable assets and infrastructure
3. **product development**: the development and sustainment of an individual product

The responsibilities associated with each managerial role help identify goals and issues that need to be addressed with information derived from software measures. The responsibilities of each managerial role are defined below.

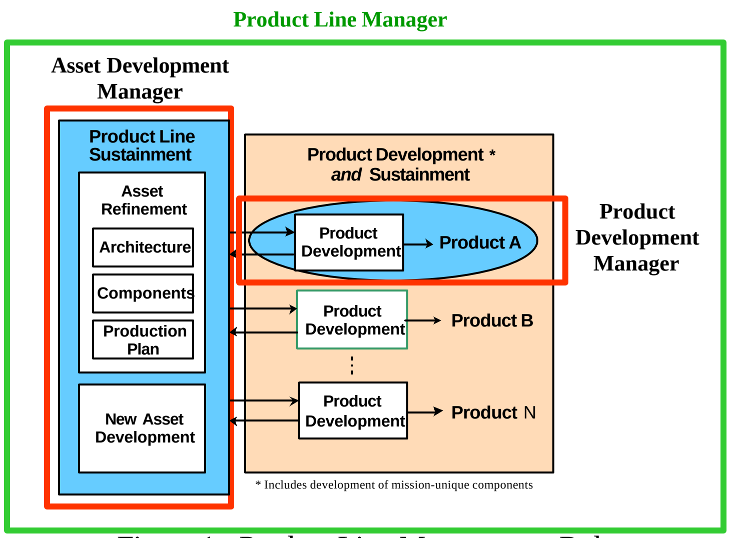

Figure 1: Product Line Management Roles

A **product line manager** is responsible for the overall business enterprise and, therefore, is concerned with the entire set of past, current, and future products that comprise the product line. The extent of a product line is generally characterized in terms of the targeted market or mission of the product line organization, identified by representative customers and the type of problem that these particular products are meant to solve. Furthermore, the product line manager needs to demonstrate the benefits associated with this particular approach to software development.

An **asset development manager** is responsible for the set of core assets and associated infrastructure that enables the streamlined development of the products in a product line. It is expected that all products in the product line will be built as instances of the product family (that is, using the provided assets and infrastructure). The asset development manager needs to provide high quality assets on a timely basis to the product development projects.

A **product development manager** is responsible for a single product. A product is the outcome of a project tasked to deliver a solution to a single customer or market problem. A product is an instance of the product line and reflects the scope of the product line and its architecture. The product development effort may be constrained in the development approach used in that it is expected to make use of product line core assets to produce their products. By doing so, it is expected that the overall goals of the product line will be realized. The product development manager is focused on providing high quality products, on time and within budget, while adhering to the product line processes and contributing to the achievement of the product line goals.

In summary, the management of a software product line encompasses traditional project management concerns as well as issues such as scoping the product line and management of core assets. In particular, the role of the product line manager is to bring business- or enterprise-level concerns into the software development arena. These roles provide the context and define the information needs to be supported by software measurement.

3. Policy Link and Issues

This author has been unable to identify any specific DoD policies regarding software product lines.[1] However, current DoD interest in software product lines may be the result and evolution of its earlier interest in software reuse. Similar to the anticipated benefits of software reuse, software product lines offer a strategy to achieve better, faster, cheaper software. Two examples of DoD investment in software reuse were the Defense Information Systems Agency (DISA) Software Reuse Initiative and the STARS project.

The DoD Software Reuse Initiative had the following goals:

- Achieve demonstrable savings in software life-cycle cost.
- Increase productivity.
- Improve the quality and reliability of software intensive systems.
- Enhance system interoperability.
- Provide earlier identification and improved management of software technical risk.
- Shorten system development and maintenance time.
- Remove barriers to effective use of software developed for government, commercial and foreign markets [CIM 96].

The STARS project also sought to improve quality, cost, and cycle time through reuse. "With the ever increasing cost of developing and maintaining complex command and control software systems the DoD community has consistently chased silver bullets, with little or no success. The Office of the Secretary of Defense recognized this situation and established the [STARS] program under the Advanced Research Projects Agency (ARPA) that evolved the Megaprogramming concept to address the crisis. The three tenets of Megaprogramming are:

1. domain-specific reuse
2. process-driven development
3. advanced software engineering environments (SEE) [Bristow 95].

Domain specific reuse is a concept closely related to software product lines. The following paragraphs taken from the DoD 5000.2 suggest a software product line approach is highly consistent with current acquisition policy. (Italics have been added for emphasis.)

5.2.6 Software Management
The PM shall manage and engineer software-intensive systems using best

[1] This statement is based on a search of various DoD web sites and a review of SEI workshop reports and technical notes that involved DoD participants or suggested a lack of established policy in this area.

processes and practices known to reduce cost, schedule, and performance risks.

5.2.6.1 General
The project manager shall base software systems design and development on systems engineering principles, which includes:

Developing *architectural based software systems* that support open system concepts; exploit COTS computer systems products; and allow incremental improvements based on *modular, reusable, extensible software.*

Identifying and exploiting, practicable, government, and commercial software *reuse opportunities* before developing new software.

Structuring a software development process that recognizes that *emerging requirements will require modification* to software over the life cycle of the system. In order to deliver truly state-of-the-software, this process should allow for periodic software enhancements.

While these principles do not directly call for a product line approach to developing software, the criteria they establish are well aligned with the benefits attendant to software organizations using a product line approach for software development and sustainment. In summary, past DoD initiatives in software reuse and the goals of current acquisition policy make a software product line approach relevant to DoD software engineering policy.

4. Information Needs

The following topics regarding a software product line should be informed with software measurement and analysis:

- the decision to adopt a software product line approach to software development
- the ongoing operation and the overall performance of the software product line

Following is brief a discussion of the analytical challenges associated with the decision to adopt a software product line. The primary focus of this paper, however, will be measures associated with the ongoing operation and overall performance of the software product line.

4.1 The Decision to Adopt

The topic suggests the need for an economic analysis of existing and planned software products in order to estimate the *economy of scope* that might be realized. This is a fundamental input to the decision as to whether a product line approach would be justified. Economy of scope refers to the extent an organization can leverage commonality across its software products to reduce cost while increasing the variety of products produced and supported [Withey 96]. This variety, however, is bounded by the scope of the product line and built upon a common architecture and set of core assets. Through the product line's ability to engage in large grained reuse of software assets, it can reduce cost, reduce cycle time, and improve quality simultaneously. However, the capability to do this requires investment in the product line infrastructure and core assets. Typically, this investment is large relative to custom software development. Therefore, an analysis must be performed to identify a break-even point. (See Withey [96] for a general discussion on this point.) The organization can then make a decision as to whether the conditions or assumptions associated with the attaining the break-even (and subsequent positive return) are feasible.

This question was addressed during a recent workshop held at the first Software Product Line Conference (August, 2000) and by participants in a follow-up email discussion. During these discussions, it was mentioned that often organizations first exploit commonality in products through ad hoc reuse and then mature this practice. However, little formal economic analysis associated with the adoption of a product line approach is done. For organizations wishing to more purposely explore the question of whether to adopt a product line approach, the following analyses were suggested

- Look for clones of software artifacts as evidence of a latent product line.
- Develop a matrix of products versus features.

Clones or features found in many products (existing and planned) are indicators of areas to explore as forming the basis for a product line.

The results of these analyses can be used to inform a more formal economic analysis that will compare the savings and costs associated with the envisioned software product line with the expected costs of software development using its current process.

For a product line to achieve its promise of better, faster, and cheaper software, it must carefully define its scope of operation or development. The scope defines the commonality that every member shares and the range of allowable variants, including the ways in which they vary from each other. For a product line to be successful, its scope must be carefully defined and managed. If an attempt is made to encompass product members that vary too widely, then the core assets will be strained beyond their ability to accommodate the variability, economies of production will be lost, and the product line will collapse into the traditional, one-at-a-time product development effort. If the scope is too small, then the core assets might not be built in a generic enough fashion to accommodate future growth, and the product line will stagnate. This decision and the follow-on implementation of the product line are crucial for enabling the product line to realize its intended benefits.

This area is not well established or validated in terms of measurement and analysis. Application of theoretical concepts such as economies of scope, and, to a lesser extent, options theory, seem to be accepted by the research community associated with software product lines. Some applied research is underway [Schmid 00] and some vendors [e.g., Debaud 00] have developed tools to facilitate this type of analysis.

4.2 Ongoing Operation and Performance

The information needs associated with ongoing operation and performance of the software product line result from an understanding of the managerial roles described earlier. Successful operation of the product line can be characterized in terms of the following dimensions

- **performance**: characterization of the extent to which development projects actually meet relatively aggressive cost, schedule, and quality objectives targets as compared to traditional product development within the organization. It is expected that a product line will set relatively aggressive goals.

- **compliance**: Characterization of the extent to which development projects utilize the processes, practices, and standards designed to leverage and reuse large-grained, common assets.

- **effectiveness**: Characterization of the extent to which the product line meets its goals and those of the overall enterprise in the marketplace and with customers.

In turn, the dimensions of success can be used to identify a set of measures useful to the enumerated managerial roles. The indicators and measures in the following table are not to be construed as an exhaustive set; rather, they are ones that we believe have elements unique to a software product line.

Objective	**Product Line Manager**	**Asset Development Manager**	**Product Development Manager**
Performance	• Total product development cost • Productivity • Schedule deviation • Time to market • Time spent on life-cycle activities • Trends in defect density	• Cost to produce core assets • Cost to operate infrastructure • Schedule deviation • Defect density in core assets • Number and type of artifacts in asset library	• Direct product cost • Defect density in application artifacts • Percent reuse
Compliance	• Mission focus • Process compliance	• Mission focus • Process compliance	• Process compliance
Effectiveness	• Return on Investment • Market satisfaction • Market feature coverage	• Core assets utility • Core assets cost-of-use • Percent reuse	• Customer satisfaction

Table 1: Product Line Indicators and Measures [Zubrow 00]

In the following sections we discuss each of the proposed measures for each of the management roles.

5. Status of Measurement for Software Product Lines

The indicators and measures listed in Table 1 were selected to address the information needs of the three management roles associated with a software product line. This is not to suggest that all the indicators and measures in the table have been validated through experience. Indeed, based on my analysis and investigation, it appears that little work has been performed to specifically address software measurement in the context of software product lines. Measurement experience in a product line context is limited by the relatively few number of organizations actually practicing a product line approach to software development as described; hence, there is little concrete evidence as to what measures have been found to be useful and practical. Yet there are relevant sources of experience upon which we can draw. As noted below, there is an extensive literature on software reuse that is relevant. The companion paper in this series on measures for object-oriented (OO) development contains measures that could be useful in the product line context.

Much of current software measurement practice related to software project management also applies to software product lines.[2] Product development and asset development efforts will entail many of the same issues as traditional software projects. As noted in the "Performance" row in Table 1, many typical measures of project performance apply to projects for product and asset development. In some instances, the measures require modification to accomplish their purpose in the product line context. Key differences between a product line approach and custom software development are the following:

- the need to manage and measure the product line in addition to individual product development projects
- the need to measure assets and asset development in addition to product development

This leads to a difference in goals that should be addressed with software measures.

"The metrics needed to manage the reuse business are more complex. While advanced companies are doing fairly well at measuring traditional software engineering projects, using metrics to run a reuse business is something new that management needs to think about….Detailed measurement is crucial to ensure that the overall reuse business goals are met" [Jacobson 97].[3]

As an example, for measures of total product development cost, some authors (e.g. Poulin 97; Gaffney 89) have developed formulae for pro-rating the cost of developing core assets to product development projects.

Using a GQM-based approach, DISA identified core reuse measures for in the following areas: size, effort, schedule, quality, reliability, reuse, and reuse inhibitors. The intent of these measures was to demonstrate the "the level of success of reuse employment within the DoD [CIM 96]."

Based on the DISA experience as well as Table 1, work in the following areas would seem to be relevant and useful with respect to operationally defining the indicators and measures of potential practical value to a software product line:

5.1 Product Line Operation

- Project management measures are well established in practice and well described in the literature.

- Product line management measures have parallels in general enterprise management, but a software product line introduces some new challenges.

Measures of software reuse provide indications as to whether the organization is in fact implementing projects in a product line fashion. They also provide insight into the utilization of core code assets. In particular, this literature gives much attention to the economics of developing assets for reuse as compared to crafting components for use in a single product. There are many sources and proposed measures. The measures in this area are relatively mature. These measures, however, are not sufficient by themselves with respect to a software product line. For instance,

[2] This statement is supported by the author's literature search as well as comments from experts in the software product line field.
[3] While I believe Jacobson's observation is correct, it speaks more to the implementation and utilization of software measurement than to the measures themselves. It is reflective, however, of the dearth of practical experience available on software measurement in a software product line.

the application of these measures in the context of large grained reuse (e.g. reuse involving several related software work products such as requirements and corresponding designs, code, and tests) has not been investigated.

5.2 Architectural Conformance

- Measures of software architecture could be used to compare product designs with the product line architecture to assess conformance [Avritzer 98]. They may even be used to determine whether a particular product should be developed as part of the product line or as a custom development project. Measures in this area are just emerging[4]

5.3 Core Asset Quality

- NIST [Salamon 94] published a paper on Quality Characteristics of Reusable Software. While this paper lists many potential measures, the status of these measures with respect to practical application and validation is unknown.

5.4 Decision to Adopt and Product Line Scoping

- Measures of product line scoping, an emerging area of study, is an economic analysis conducted to establish the scope and viability of a product line. This is most directly related to establishing and revising the scope of the product line. Additionally, such an analysis would be an input to the decision as to whether adopting or establishing a product line makes any business sense.

- Applications of economic analyses such as economies of scope, real options theory, amortization, and NPV can be used to help establish the investment and adoption strategy for a software product line. These analyses would also be most relevant to decisions on establishing and revising the scope of the product line. It is the judgment of some experts in the field (e.g., Lim in personal communication) that such analyses are not widely practiced. Yet guidance on how to perform these analyses has been published [Lim98; Poulin 97; Reifer 97].

Given the status of these areas of measurement and analysis, the following sections discuss and describe potential indicators and measures. The intent is to provide suggestions for getting measurement activities established within the software product line, not to be an exhaustive accounting of all potentially useful measures.

6. Indicators and Measures for Product Line Management

Management of the overall product line effort is the role for which the most writing and investigation has been done with respect to software measurement. It is with respect to this role that most of the metrics and models associated with a product line approach to software development have been developed.

[4] For instance see the paper by Elaine Weyuker on predicting project risk from Architecture Reviews.

6.1. Total Product Development Cost

Total product development cost measures the engineering costs incurred by the product line organization to create a new software product. It consists of two components

1. the direct product development costs incurred by the product development group
2. a prorated share of the asset development costs

The costs of the product development group should be actual expenditures captured by the effort tracking system. If significant costs are incurred to purchase common off-the-shelf software (COTS) components, these too should be included in the product development expenditures. Various schemes [DISA 95; Gaffney & Durek 89; Poulin 97; Poulin ND; Reifer 97] have been proposed for prorating the asset development costs to a specific product development project/product. For the purposes here, we propose a simple formulation of:

$$AllocatedCost = \frac{AnnualAssetDevelopmentCosts}{NumberofNewProductDevelopmentProjects}$$

which can be used to determine the amount to be allocated to product development projects. Annual Asset Development Costs could initially be based on the budget for an asset development group if actual expenditures are not available.

Depending upon the rate for new product releases, this metric could be tracked on a quarterly basis. In this case, costs associated with actual core asset and product development projects completed during the quarter would be tracked. Note that there will necessarily be a gap between expenditures on core asset development and the availability of the assets produced for inclusion in product development projects. The significance of this lag should diminish as the product line matures.

Total product development cost should decrease as the product line matures. Initially, it may be higher than traditional product development costs since the costs to establish the core asset infrastructure will be spread over a relatively small number of projects. Over time, however, it is expected that relatively less effort would go into the infrastructure and that the assets produced would be highly leveraged by product development projects. As a result, the total cost to the organization for new products should be reduced.

As an alternative to the preceding formulation, some percentage of core asset development could be allocated to future projects. For example, the overall product line business plan may call for 25 new products to be released during its first 2 years of operation and another 75 during the next 3 years. In addition, the plan may call for $1M to be invested in developing core assets during that 5-year period. Ignoring for now the time value of money, $10,000 would be allocated to each the cost of each product development project to cover the investment in core asset development.

Whatever method is selected, it should be based on the product line business plan and follow accepted accounting procedures for cost recovery on investments with no depreciation in asset values. Also, costs allocated to future projects should be done at an inflated amount compared to costs allocated to current projects due to delay in use and hence, the likely increased value of the earlier investment to develop the core assets.

Note that as formulated above, this metric does not seek to measure how the core assets are used. It assumes that a trend towards reduced development costs is due to reuse of core asset. If such a trend is not observed, this metric provides little insight as to why. This metric could be evolved over time to become more precise and insightful. For instance, the prorating of asset development costs might use a formula that evenly apportions non-code costs across all projects and code products based on historical and projected use.

6.2 Productivity

Measuring and tracking productivity provides insight as to whether the product line approach enables the organization to develop more product with the same or fewer resources. Productivity is defined to be the ratio of the amount of product or output of the organization relative to the resources consumed to produce it. There are several options for measuring output. Output can be measured in terms of number of products fielded, number of features delivered in the products, or some measure of the size of the products such as lines of code (LoC) or function points (FP). Resources consumed will most likely be represented by effort expended. Effort may be measured in terms of hours and should include not only effort expended by product development but also effort expended on asset development activities. It is suggested that the organization initially compute productivity based on a cumulative accounting of products delivered and effort expended. This simplifies the accounting with respect to aligning asset development costs with the delivered products.

To compute a productivity figure, an organization would measure the amount of product delivered since the inception of the product line effort. This would then be divided by the cumulative effort of the corresponding application development projects and the total investment in the development of core assets during the same time period. Tracking productivity for the product line can be done on a quarterly basis. More frequent reporting may not be worthwhile unless the product line is releasing products on a monthly basis. That is, the update of the productivity measure should match the flow of products released by the product line.

This measure of productivity reflects on the entire product line enterprise. Over time, it should trend towards higher productivity as the investment in core assets and infrastructure are spread across an increasing number of products and as the cost of creating a product using the assets and infrastructure decreases. To determine whether the rate of improvement in productivity is acceptable, the productivity needs to be compared against some expectation. Therefore, it is useful to estimate productivity as part of the overall product line plan, even if only at a high level. A high level estimate could be made using the budgets for core asset and infrastructure development, the budget for product development, and the estimated size of the products to be released during the year.

This measure indicates one dimension of overall performance, but does not reveal what is driving change in performance. Additional detailed measures of the performance for developing core assets, their use in product development, and the performance of product development teams is needed to gain insight into the underlying factors driving productivity trends.

6.3 Schedule Deviation

The product line manager is responsible for the delivery of all product line related products, those produced by both asset development and product development. Since less new development is required to produce products as the product line matures, the product line should enable meeting schedules more reliably as it matures. Schedule deviation for the product line manager is the sum of the variance of all product schedules. This measure should be calculated separately for completed projects and those underway. Schedule deviation reflects both on the accuracy of the planning and the ability of the project to execute the plan.

Variants on this measure include weighting the variances by the planned amount of effort to be expended by the project. Additionally, the absolute value of each schedule variance can be summed to prevent projects that are ahead of schedule from offsetting those that are behind schedule. If the organization uses an earned value management system, schedule measures such as the schedule performance index and schedule slip could also be calculated.

6.4 Time to Market

This measure captures the capability of the organization to deliver products and features faster.[5] This measure is based on the duration or calendar time of completed projects and the functionality they deliver. To measure the duration of the project the organization must establish operational definitions of when projects begin and end. As an example, the start of a project is marked when approval to undertake the project is received and work to build the product has commenced. The end of the project could be marked by the completion of acceptance testing or by product release. If the product line organization delivers products containing largely varying amounts of functionality, it should either normalize the measures or bin similar projects into relatively homogenous groups. To normalize the measure, we suggest using the organization’s standard size measure (e.g., LoC, FP, number of requirements). The measure should be reported as the duration to deliver a given amount of software (e.g., months/KLoC). Grouping may be accomplished by creating categories of small, medium, and large.

Time-to-market can be tracked and reported on a periodic basis (e.g., quarterly, annually) determined by the number of projects in the product line and their spacing in terms of completion. Product development project duration should be analyzed in terms of both its central tendency (i.e. average, median) and dispersion (i.e., variance, range). This measure should be also used for product planning, especially estimating overall project schedules.

6.5 Trends in Defect Density

Defects in delivered products reflect their quality. As defects are removed from core assets, the quality of products developed using the product line approach should increase. All defects in delivered products should be tracked to determine if product quality is improving. To compute this measure, defects reported by customers are divided by the size of the delivered product. Only unique defects reported against a product are tallied. A time period must be set for accumulating and reporting the defect count. The time period depends upon the use and complexity of the products, but it should be long enough to allow users to thoroughly test the product’s features and capabilities. It may be as short as three months or as long as one year. Trends in the average defect density per reporting period for all products should be plotted. In addition, the variation in defect density as represented by the individual values for each product should also be plotted.

[5] We recognize that for various business reasons, the pace of new product introduction may be managed to be slower than the capability of the product line to deliver new products.

6.6 Mission Focus

Mission focus measures evaluate the degree to which products produced by the product line organization fit within its defined scope. These measures attempt to ensure that the product line maintains a coherent focus consistent with the strategic objectives delegated to the product line organization by higher management. Initially, this is a requirement for the product line for which an objective indicator may not exist. Therefore, as the product line is beginning, the data for this measure could come from surveys of the organization's managers that asks for their judgments regarding whether the products being produced by the product line are, in fact, consistent with its mission and scope. Once the product line and its corresponding architecture are more established, another indicator would be to track reuse of early life-cycle artifacts such as requirements and designs. It is assumed that products using a significant proportion of these types of core assets in their development are more aligned with the product line than those using relatively fewer of these types of core assets. Mission focus should be reported on a periodic basis (e.g., quarterly). As mission focus improves, other measures such as productivity and quality should improve due to the increasing use of core assets.

6.7 Process Compliance

Establishing a product line approach to software development will yield few results if it is not followed and implemented. Process compliance measures capture the degree to which products are produced using the product line process. The data for this measure could come from process audits conducted by the organization’s software quality assurance function. For the purpose of the product line manager, a simple count of the number of deviations discovered, resolved, and open across all projects is an adequate starting point. Both core asset development and product development projects should be included. These data should be reported in concert with the periodic management reviews of the product line operation. Over time, additional analysis of these data can be used to improve the product line development process. These analyses could look at differences in deviations between asset development projects and product development projects, variation in deviations by process or life cycle activity, or type of project (e.g., maintenance, enhancement, new product).

6.8 Return on Investment

Estimates of the return on investment (ROI) are often made to support a change in business practices. ROI is computed as the ratio of the estimated savings (or returns) for each dollar invested By estimating the ROI for various alternatives for improvement, the alternatives can be compared and ranked in terms of their likely economic benefit to the organization. This applies to the adoption of a product line approach to software development.The challenge for this measure is to clearly and objectively estimate the savings and to clearly identify and estimate the investment, the costs to transition to a product line approach and the costs to develop products using the product line approach. Savings need to be estimated with respect to the current way of producing software. This implies that the organization tracks project costs. To estimate savings, the organization needs a model of how the product line will operate and be used within the organization. It needs to set a time period during which a given volume of products will be produced.

The ROI formula for a given volume of product produced is

$$ROI = \sum \frac{Costt - Costp}{Costi}$$

Where $Costt$ is the cost of traditional software development, $Costp$ is the cost associated with using the product line approach, and $Costi$ is the cost of establishing the product line. The differences in development costs are summed across the expected number of projects.

This comparison is simplest if the comparison is done for the product line operation as a whole (asset development and product development) with a comparable amount and mix of product produced using the organization's current approach. Costs to establish the product line include non-recurring start up costs, such as developing a concept of operations, product line architecture, and the acquisition of tools unique to the operation of the product line.

Furthermore, these costs can be amortized across the expected life of the product line. This will avoid charging the total cost of these assets and the infrastructure solely to the early projects. The time horizon or usable life of the core assets selected and the method for allocating costs across this time period should be consistently followed for all analyses utilizing these data. This includes analysis of total development costs as described above.

While ROI is often computed to compare alternative courses of action prior to implementation, it is important to continue to monitor it to measure whether the anticipated return is actually being achieved. Therefore, the measurement system employed must capture the needed data from both core asset and product development projects. Also, the costs to create infrastructure components to support the overall product line must be captured.

6.9 Market Satisfaction

The product development manager should measure the satisfaction of customers purchasing products. This might most easily be accomplished through a survey. In addition to general satisfaction, the survey should address dimensions of satisfaction such as reliability, functionality, usability, and value (the relation of these attributes to price). Trends in satisfaction value for the general rating and its constituent dimensions (e.g. reliability, usability) should be analyzed. Similarly, variation in the responses from customers should be analyzed to help identify potential improvement opportunities. For instance, certain products or customer segments may reveal trends that the product line can leverage to its benefit. A supporting measure is the number of complaints that are received that relate to the product. (See the discussion on defect density above.) This number should be normalized by volume of product delivered if the volume has been changing over time.

6.10 Market feature coverage

This measure would capture the extent to which the features currently available by the product line cover all features identified as relating to the target market. This could be expressed as a percentage. The key towards assessing progress, and hence the utility of the measure, of the product line is the identification of features that are relevant to the market. This set should be used to establish the denominator of this measure. Additionally, it should be reassessed periodically.

7. Indicators for Asset Development Management

These indicators are to be used to characterize the performance of projects to produce core assets, the value of these assets to product development projects, and the management of the asset repository. They do not cover all of the indicators that an asset development manager might use to actually manage a project; additional measures or variants on the measures might also be used.

7.1 Cost (Effort) to Produce Core Assets

The cost to produce the various core assets should be tracked. Not only are these costs important from the typical project control or management perspective, but they are a major input into tracking the success of the product line approach. Additionally, understanding how the cost of designing for reuse compares to traditional product development costs is a key component for deciding what components are worthwhile to construct for reuse. It is expected that developing assets for reuse will be more expensive than developing them for a single specific use. The degree of additional expense needs to be monitored and controlled. Note that Poulin (97) and Boehm (CoCoMo II) assume that the cost to produce these assets may run 1.5 to 2 times higher than producing the same functionality for a one-time use. On the other hand, the use of such assets runs about 1/5 the cost of developing the functionality.

These costs resemble typical software project costs. These measures should track the development of both functional (code) components as well as the development of non-code assets such as domain requirements, architectures, and user documentation. These costs should include all direct labor costs and be associated with the specific asset that is made available to application developers.

7.2 Asset Development Schedule Deviation

Because core assets are planned to be used in multiple subsequent product development projects, their availability as planned is crucial to the efficient and effective operation of the product line as a whole. The performance of the asset development project against its planned schedule should be tracked. Furthermore, this schedule needs to be visible to product development projects that are dependent upon the core assets under development. As revisions to the schedule occur, this history should be retained to serve as an input for future improvement in estimation and scheduling. Typically, schedule deviation is computed as the duration between the planned and delivered dates, where planned dates are based on projected needs of product development projects. As with cost, it is important to track this measure for the development of both code and non-code assets. Critical to the functioning of the product line is development of its architecture. Slipping the schedule in this non-code asset would impact multiple product development projects.

7.3 Defect Density of Core Assets

It is extremely important to manage the development/acquisition of the product line’s core assets because core assets will impact the quality, reliability, and cost and schedule of every derivative product development. The success of the product line rides on the quality of the core assets and the capabilities that they offer. This measure indicates the quality of the core assets used to produce applications. To compute this measure, defects reported by product development projects and customers are divided by the size of the core asset. Only unique defects associated with core assets are counted. The defects need to be attributed to the core asset in which they originated. A time period must be set for accumulating the defects from customers, as discussed above. Trends in the average defect density for core assets and in its variation should be plotted.

7.4 Standards and Process Conformance

Measures of standards conformance capture the degree to which core assets and tools are being produced in accordance with the organizational product line standards. These measures provide insight into the extent to which product development projects comply with the organizations product line processes and practices. Such data are useful for establishing a context for interpreting other performance data associated with the product line. They also provide insight into the adoption of the product line practices by product development projects during product line start-up. The data for this measure could come from process audits conducted by the organization's or product line's software quality assurance function. For the purpose of asset development management, a simple count of the number of deviations discovered, resolved, and open across the asset development effort is an adequate starting point. These data should be reported in conjunction with the periodic management reviews of the product line operation.

7.5 Core Assets Utility

Asset development management should track the extent to which core assets provide value to product development projects. This accrues cumulatively across product development projects as well as with respect to individual projects. This provides insight into the extent to which core assets are able to satisfy the goals of the product line as well as helps determine whether the assets in the asset library are the assets actually needed by product development projects. Knowledge as to what core assets are used to create products through product development projects should be available from the configuration management system.

One measure of the utility of core assets is the impact their use has on product development project performance. This can be measured in terms of cycle time, cost and quality. This measure requires that an estimate be made as to what the performance of the product development project would have been had the core assets not be available. These estimates also support computation of the overall return on investment of the product line, as was discussed earlier. This measure would then be represented in a cumulative manner for all products developed as well as what could be expected for individual projects (e.g., median and range or mean and standard deviation).

If the data required for this measure are not available, an initial metric would be percent reuse [DISA 96]:

$$\frac{SizeofCoreAssetsUsed}{(SizeofCoreAssetsUsed + SizeofNewandModifiedApplicationArtifacts)}$$

In addition to percent reuse, a count of the number of uses of each core asset divided by the number of products in the product line can provide insight into those assets that are commonly used and those that are less frequently used. This will help determine whether the assets in the asset library are the assets actually needed by product development projects. Ferri, Pratiwadi, Rivera et al (1997) discuss several metrics (e.g., code profiles and reuse growth factor) that provide further insight into the use of core assets.

7.6 Core Assets Cost-of-Use

Core asset management needs to be cognizant of the costs incurred by product development projects in order to use core assets. If the costs become too large, product development projects may seek other ways of satisfying product requirements. From the perspective of the product development project, the costs they incur are those associated with (1) identifying appropriate core assets, (2) understanding how to apply them to the project, and if need be, adapting them for use, and (3) finally integrating and testing them. Costs in this category would primarily be driven by the effort spent by engineers to find, understand, and tailors assets for use on their projects.

If the effort tracking system does not support this level of effort tracking by activity, a study could be done as the basis for estimating the cost. Core assets cost-of-use should be tracked across all projects and for each asset type. The finer grained the data, the greater the insight core asset management will have into how this cost can be reduced.

In addition to the costs incurred by product development projects, costs may be incurred to create and implement a repository that facilitates the management and use of core assets.

8. Indicators for Product Development Management

These measures are to be used to characterize the performance of product development projects to produce products. They do not cover all of the measures that a product development manager might use to actually manage a project; additional measures or variants on the measures might be used. For the most part, this role requires few metrics beyond those typically used for software project management. On the other hand, the application project setting is the source of much data on the performance of the product line. Therefore, there must be a measurement system that provides for the collection of the needed data.

8.1 Direct Product Cost

These costs resemble typical software project costs, and should include all direct labor costs incurred while producing (i.e., requirements analysis, design, construction, and testing) the product. Application development management should monitor the relative proportion of project costs going to produce application specific code. The remaining costs can be interpreted as the cost to integrate core assets into the product. Over time, it would be expected that the former would decline as a proportion of the latter and that the latter would decrease in absolute terms as the organization gains experience and the quality of the components improves.

8.2 Defect Density of Application Specific Code

The product development management role needs to understand the quality of the code it is producing as well as the quality of the assets it is utilizing. To compute this measure, defects reported by customers are divided by the size of the application specific code in the product. During repair, each defect needs to be attributed to either application specific code or a core asset. Only unique defects are tallied. A time period is usually set for accumulating the defects from customers, as discussed above. Trends in the average defect density for application-specific code and in its variation should be plotted. This information can be used to gauge the performance of the application development software processes.

A related use of the data is to track trends in the types of defects discovered by users. Such information is useful for identifying areas for process improvement.

8.3 Process Compliance

The product development manager needs to monitor compliance with processes associated with the operation of the product line. Since it is likely that the processes will be incorporated into the estimating and planning of product development projects, failure to comply with these processes may put the project at risk. As described above, reports from audits and reviews should be provided to the product development manager. Furthermore, these reports should make it easy for the manager to assess the degree to which the project is adhering to the product line concept of operation.

8.4 Percent Reuse

Planned reuse plays an important role in estimating the cost and schedule for product development projects. Product development management should track which core assets are used in the product and the extent to which each is used. Knowledge as to what core assets are used should be available from the configuration management system. The use of core assets in all work products, not just code, should be tracked. Failure to achieve planned reuse levels may signal performance trouble for the product development project and may also trigger a causal analysis by core asset management. For instance, the product development project may find itself creating functionality from scratch when it planned to only integrate a core asset with the desired functionality. Hence, there would be a likely impact on cost and schedule.

Percent reuse can be computed as follows:

$$\frac{SizeofCoreAssetsUsed}{(SizeofCoreAssetsUsed + SizeofNewandModifiedApplicationArtifacts)}$$

As the above example illustrates, a product line reuse of assets goes beyond code. A size measure such as “pages” or “requirements” will need to be established to compute this measure. Additionally, an overall summary measure can be created by converting assets and artifacts to a common measurement unit such as lines of code.

Lim (1998) cautions that such a measure can overstate the percent reuse, and hence, any measures derived from such a measure like costs avoided. The reason for this is that the reused asset may include functionality not required for the product being developed.

8.5 Customer Satisfaction

Product development managers need insight into how well the products produced by their projects satisfy their intended customers. As products are delivered to customers, the product development manager can ask for feedback via a customer satisfaction survey. This survey can ask questions regarding whether the product satisfies requirements and can also address satisfaction with respect to how well the project met its cost and schedule commitments. These data should be retained and analyzed cumulatively as more products are produced.

9. Recommendations for next steps

The following recommendations are made for further study into this area:

1. Clarify DoD interest and position with respect to product line development approach. As discussed above, many existing software measures (i.e., traditional project management and software reuse) are available for various aspects associated with the operation and performance of a software product line. These may or may not be adequate to the needs of the DoD.

2. Identify and interview leaders in the use of software product lines. Given that a software product line approach to software development is relatively new, the best approach to investigation is to conduct in-depth case studies with successful organizations. Much could be learned from their practices.

3. Summarize the case studies to identify the enablers and obstacles to adopting a product line approach. Take specific note of measurement and analysis practices. For instance, what are the key elements and analytical techniques used to develop the business case supporting a decision to adopt a product line approach?

4. Conduct empirical research on areas deficient in measures and/or analytical techniques such as architectural conformance.

References and Bibliography

[Avritzer 98] Avritzer, Alberto and Weyuker, Elaine J. “Investigating metrics for architectural assessment,” IEEE, 1998, pg 4-10.

[Bass 97] Bass, Len; Clements, Paul; Cohen, Sholom; Northrop, Linda; & Withey, James. *Product Line Practice Workshop Report* (CMU/SEI-97-TR-003, ADA 327610). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. Available WWW
http://www-preview.sei.cmu.edu/publications/documents/97.reports/97tr003/97tr003abstract.html

[Bass 98b] Bass, Len; Clements, Paul; Cohen, Sholom; Northrop, Linda; Smith, Dennis; & Withey, James. *Second Product Line Practice Workshop Report* (CMU/SEI-98-TR-015, ADA 343688). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. Available WWW
http://www-preview.sei.cmu.edu/publications/documents/98.reports/98tr015/98tr015abstract.html

[Bass 99] Bass, Len; Campbell, Grady; Clements, Paul; Northrop, Linda; & Smith, Dennis. *Third Product Line Practice Workshop Report* (CMU/SEI-99-TR-003, ADA 361391). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. Available on the WWW
http://www.sei.cmu.edu/publications/documents/99.reports/99tr003/99tr003abstract.html

[Bergey 98] Bergey, John; Clements, Paul; Cohen, Sholom; Donohoe, Patrick; Jones, Larry; Krut, Bob; Northrop, Linda; Tilley, Scott; Smith, Dennis; and Withey, James. *DoD Product Line Practice Workshop Report* (CMU/SEI-98-TR-007, ADA 346252). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. Available on the WWW
http://www.sei.cmu.edu/publications/documents/98.reports/98tr007/98tr007abstract.html

[Bristow 95] Bristow, David J.; Bulat, Brian G.; Burton, D. Roger “Product-Line Process Development” Software Technology Conference, 1995.

[CIM 96] Center for Information Management, *Department of Defense Software Reuse Initiative Reporting Metrics and Measures*, DISA Joint Interoperability and Engineering Organization, 11-Apr-1996.

[Clements 99] Clements, Paul; Northrop, Linda M.; et al. *A Framework for Software Product Line Practice, Version* 2. Available on the WWW
http://www.sei.cmu.edu/plp/framework.html, August 1999.

[Debaud 00] Debaud, Jean-Marc. “The TrueScope™ Approach to Software Product Family Engineering.” Tutorial presented at the First Software Product Line Conference, Denver, Co, August 28, 2000.

[DoD 00] Interim Regulation DoD 5000.2-R -- Mandatory Procedures for Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs, 23 October 2000.
http://www.acq.osd.mil/ar/doc/dodd5000-1-int-reg.pdf

[Jacobson 97] Jacobson, Ivar; Griss, Martin and Jonsson, Patrik. *Software Reuse: Architecture, Process and Organization for Business Success*, New York: ACM Press, 1997.

[Lim 98] Lim, Wayne C. *Managing Software Reuse: A comprehensive guide to strategically reengineering the organization for reusable components*. Upper Saddle River, NJ: Prentice Hall PTR, 1998.

[Pfleeger 96] Pfleeger, Shari L. “Measuring reuse: A cautionary tale,” *IEEE Software*, July, 1996, p 118-127.

[Poulin 97] Poulin, Jeffrey S. *Measuring Software Reuse: principles, practices, and economic models.* Reading, MA: Addison Wesley, 1997.

[Reifer 97] Reifer, Donald J. *Practical Software Reuse: Strategies for introducing reuse concepts in your organization.* New York: John Wiley and Sons, 1997.

[Salamon 94] Salamon, W.J.; Wallace, D.R. “Quality characteristics and metrics for reusable software (preliminary report).” NISTIR, NISTIR 5459, May 1994.

[Schmid 00] Schmid, Klaus “Scoping software product lines.” In *Software Product Lines: Experience and Research Directions*, Patrick Donohoe (ed.), Boston: Kluwer Academic Publishers, 2000.

[USASSDC 95] US Army Space and Strategic Defense Command “Software Reuse Business Model – Technical Report” DoD Software Reuse Initiative, DISA, January 31, 1995.

[Weiss 99] Weiss, David M. & Lai, Chi Tau Robert. *Software Product Line Engineering.* Reading, Massachusetts: Addison-Wesley, 1999.

[Withey 96] Withey, James. *Investment Analysis of Software Assets for Product Lines* (CMU/SEI-96-TR-010, ADA 315653). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. Available WWW http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.010.html

[Zubrow 00] Zubrow, Dave; Campbell, Grady, and Goethert, Wolf. “Measures for Software Product Lines.” SEI Internal Paper, February, 2000.